5. System Design of a Sequential Decoding Machine,

W. A. Lushbaugh and J. W. Layland

a. Introduction. Convolutional encoding and sequential decoding are currently receiving considerable attention for use in spacecraft telemetry systems (Ref. 1). The properties of the codes and the decoding algorithm are discussed in detail in Ref. 2. Up to the present, most sequential decoding has been performed using general-purpose computers programmed according to the Fano algorithm. This article describes the preliminary design of a special purpose hardware decoder for binary convolutional codes based upon this algorithm.

b. The Fano algorithm and convolutional codes. The convolutional encoder consists of a K-bit shift register coupled with V parity check adders, each of which is connected to a distinct subset of the bits in the shift register. Typically, the code is systematic and complementary; i.e., one adder receives only the most recent bit, and this bit is connected to all adders. After each data bit is shifted into the register, the V check symbols are sampled in turn and transmitted. These V symbols form one branch of the tree code which is generated. To synchronize and block the data, the encoder is set to a known state at the beginning of each L-bit block of data, and following the transmission of these L bits, a known sequence, or tail, of at least K bits is encoded and transmitted.

The decoder for this code consists of a buffer to hold received symbols, a copy of the encoder, and equipment to measure the merit of the output of this encoder relative to the received symbol sequence. This metric is used sequentially and systematically to estimate and/or correct the local data sequence. The metric is computed branch-by-branch and compared to a threshold. Whenever a threshold violation occurs, the local data sequence is searched backwards for a probable cause and then corrected. The estimation and backward searching are controlled according to the Fano algorithm in such a way that no looping is possible; i.e., given enough time and very unfavorable circumstances, all 2^L possible local data sequences could be examined by the decoder. The basic flow diagram of the Fano algorithm is shown in Fig. 32.

c. Design aims. Although design and construction of the sequential decoder are scheduled for readiness at the launch of *Pioneer D*, a significant factor in its design is the requirement that the decoder be capable of decoding all "reasonable" convolutional codes and data formats. Reasonable has been defined to include all systematic and complementary codes of constraint length 32 or less, block size 2048 or less, having 5 or fewer symbols per branch, and having a tail of length 64 or less. The decoder must perform in a near-optimal manner and with a computation rate which is significantly faster than that obtainable by using presently available general-purpose machines under program control.

d. Computational format. In order to simplify the description of the decoder to follow, it is convenient to describe here the items which are computed by the decoder at each step. These are defined as follows:

M_p = global metric after pth branch of code

$r_{i,p}$ = ith received quantized symbol on pth branch

R_p = pth branch received data

$M(x)$ = metric value of the symbol x

$BM_p(x)$ = branch metric using pth branch data assuming pth bit was x

b_p = estimated best bit at pth branch
$b_p = 1$ if $BM_p(1) \geq BM_p(0)$

d_p = local data bit selected for pth branch

$c_{i,p}$ = ith local coder check symbol on pth branch

The branch metric, $BM_p(x)$ is further defined by

$$BM_p(x) = M_0 + \sum_{i=0}^{v-1} M(x * c_{i,p} * r_{i,p})$$

where the operation $*$ denotes multi-bit exclusive-or; i.e., each bit in the binary expansion of $x * c_{i,p} * r_{i,p}$ is the modulo 2 sum of x, $c_{i,p}$, and the corresponding bit of $r_{i,p}$.

Assume that the decoder has just completed a forward step onto the pth branch. M_p and certain other recent metric terms are available from previous computation. To determine the next step, the decoder computes the following four items:

$$M_{0,p+1} = M_p + BM_{p+1}(0) \tag{1}$$

$$M_{1,p+1} = M_p + BM_{p+1}(1) \tag{2}$$

$$\Delta_{1,p+1} = \sum_{i=0}^{v-1} [M(c_{i,p+1} * r_{i,p+1}) - M(1 * c_{i,p+1} * r_{i,p+1})] \tag{3}$$

$$\Delta_{2,p+1} = \sum_{i=1}^{v-1} [M(c_{i,p+1} * r_{i,p+1}) - M(\delta_i * c_{i,p+1} * r_{i,p+1})] \tag{4}$$

where $\delta_i = 1$ if d_p enters into $c_{i,p+1}$.

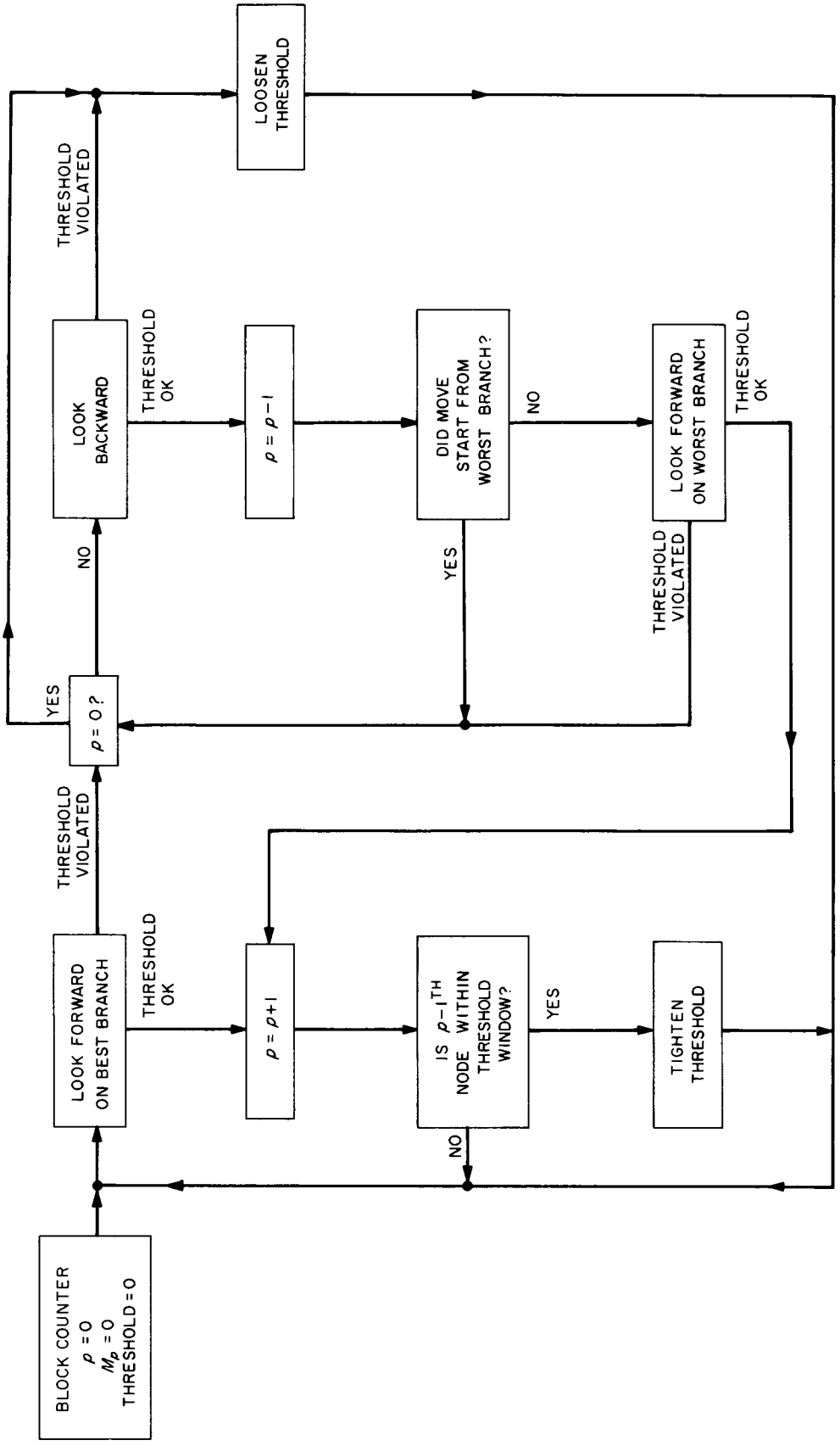

Fig. 32. Basic flow diagram for the Fano algorithm

If $\Delta_{1,p+1}$ is positive, then $b_{p+1} = 0$, and $M_{0,p+1}$ corresponds to the best branch after the pth; otherwise $b_{p+1} = 1$ and $M_{1,p+1}$ is best. $\Delta_{2,p+1}$ is used if and when it becomes necessary to examine the metrics on the pth branch after complementing d_p. Let primes denote these metrics. Then

$$\Delta'_{1,p+1} = \Delta_{1,p+1} - 2 * \Delta_{2,p+1}$$
$$M'_{0,p+1} = M_{0,p+1} - (-1)^{b_p}\Delta_{1,p} - \Delta_{2,p+1}$$
$$M'_{1,p+1} = M_{1,p+1} - (-1)^{b_p}\Delta_{1,p} + \Delta_{2,p+1}$$

and the new terms are generated with two additions instead of the $V + 1$ that were required to generate the initial terms.

Decoder action after the computation phase depends upon a subset of the stored metrics. The decoder may step backward, accept

$$M_{b_{p+1},\,p+1},$$

and step forward, or require examination of the primed terms before stepping backward, or accepting

$$M'_{b'_{p+1},\,p+1}$$

and stepping forward.

The metrics available after the forward computation phase are shown schematically in Fig. 33. Solid lines refer to metrics retained from the previous step, while dashed lines refer to metrics just computed. The computation profile for a backward step is similar to Fig. 33 with two exceptions: (1) the $p + 1$ terms are retained (along with M_p) from the previous step, and the M_{p-1}, $M_{\bar{a}_p,p}$, and $\Delta_{1,p}$, and $\Delta_{2,p}$ are computed, and (2) M_p may be less than $M_{\bar{a}_p,p}$, in which case the decoder has stepped backward to a worst branch and will execute another backward step immediately. The array of metrics available to the decoder after either the forward or reverse computation phase permit the decoder to perform in one step, via a simple decision-gating network, operations which appear to be performed in two, three, or more steps in the flow chart of Fig. 32. A functional diagram of this decision gating is shown in Fig. 34.

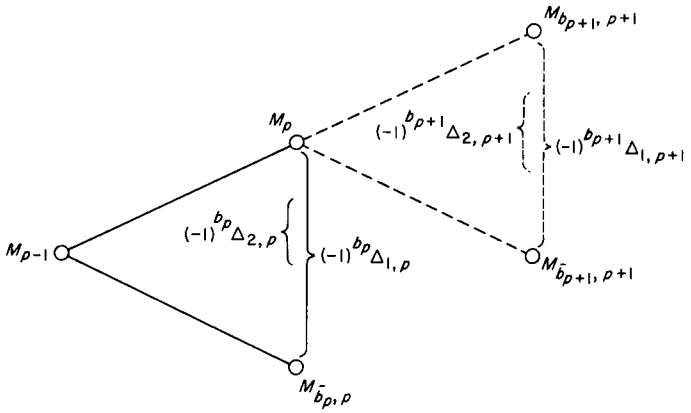

Fig. 33. Forward computation profile

e. Decoder operation. A basic block diagram of the decoder is shown in Fig. 35. The symbols as received are quantized into three bits packed into a four-bit segment of the packing buffer. When this buffer is full, it is written into the random access memory (RAM) and the next word, containing old symbols and decoded data, is read out. Transmission of the decoded data from the decoder is clocked by the received data, and it lags the initial receipt of that data by N-symbol times (where N is the RAM size in symbols). When symbols are needed by the decoding unit, they are transferred from the slow, magnetic core RAM to the high-speed buffer where they are available, symbol-by-symbol, for the computations described in paragraph *d* above. This symbol-oriented organization is a critical factor in allowing flexibility in V, the number of symbols per branch. The fourth bit assigned to each symbol in the memory is used by the decoder for two purposes. The decoded data bit from the last stage of the coder, d_{p-33}, is placed into the fourth bit of $r_{0,p}$ on a forward step, and retrieved on a backward step. Simultaneously with the forward computation of the metrics at $p + 1$, the local encoder calculates the check-digits

$$\left\{ c_{i,\,p+2} \right\}_{i=1}^{V-1}$$

as if $d_p = d_{p+1} = 0$, and places them into the fourth bit of the associated

$$\left\{ r_{i,\,p+2} \right\}_{i=1}^{V-1}$$

This precomputation allows a much longer effective time to compute each $c_{i,p}$ while still allowing full flexibility, as d_p and d_{p+1}, and only these bits, are changeable during the step following a step onto the pth branch. The stored check digits, therefore, remain valid into the past, and need not be computed on a backward step.

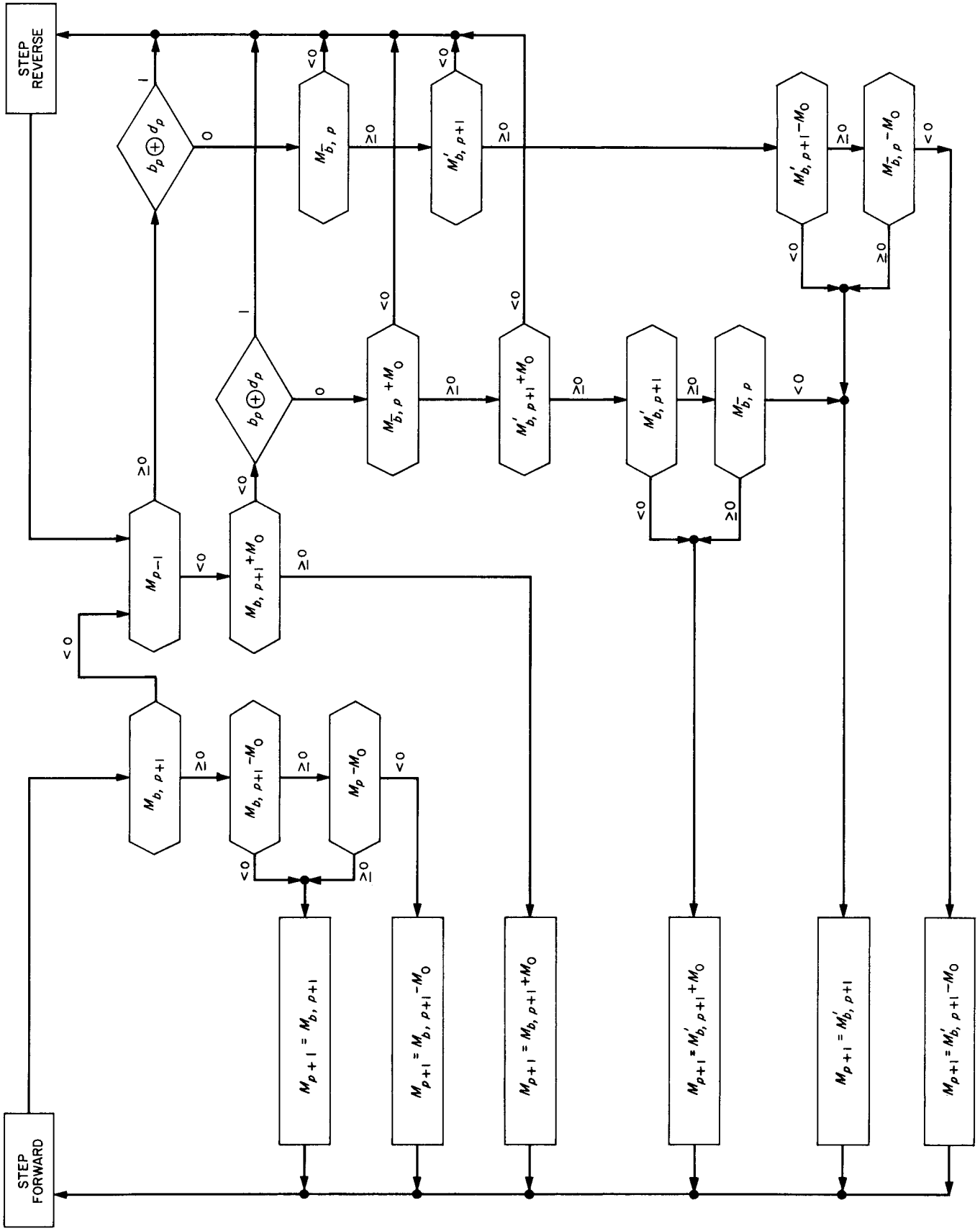

Fig. 34. Functional diagram of decoder decision gating

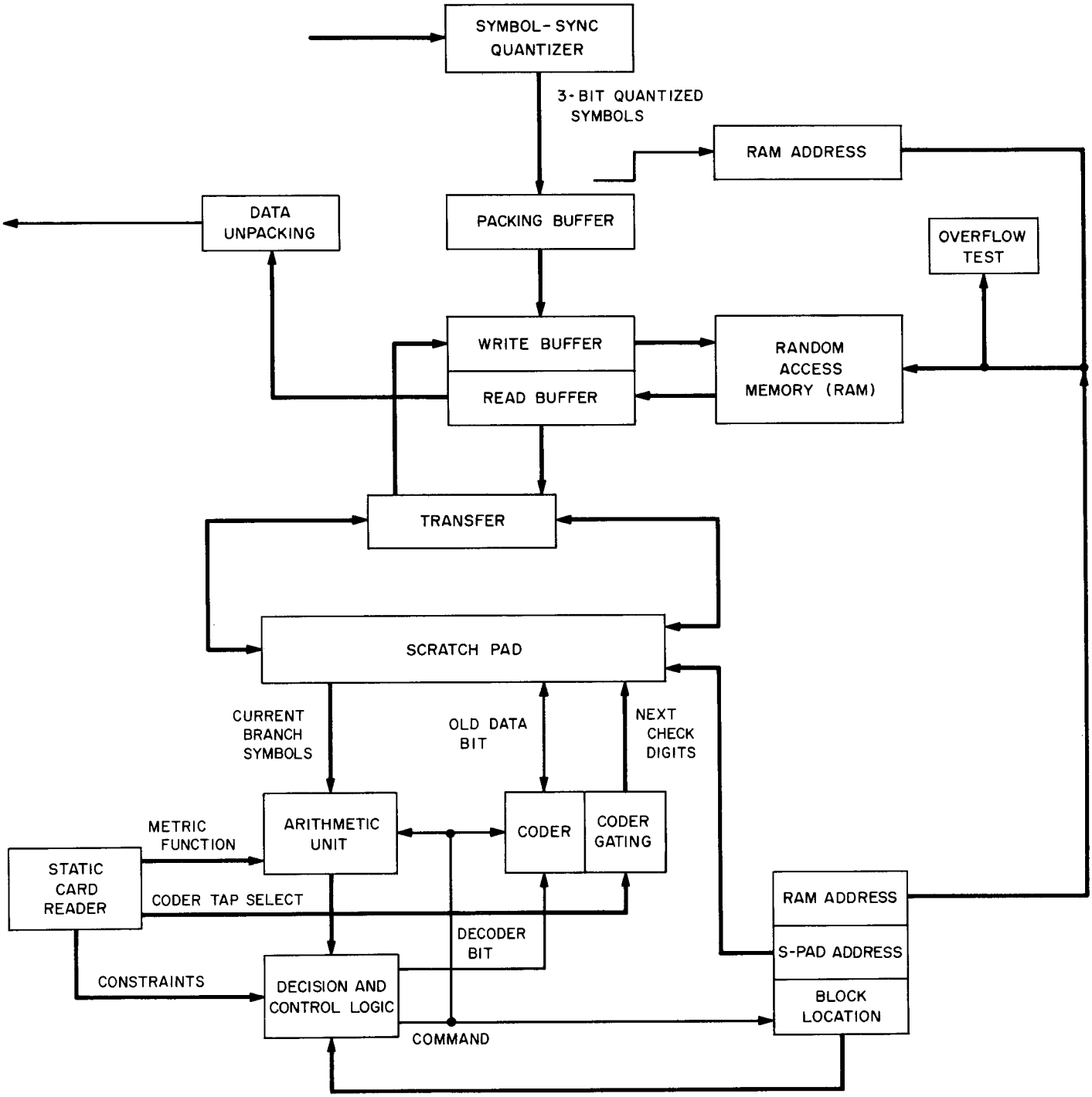

Fig. 35. Basic block diagram of sequential decoder

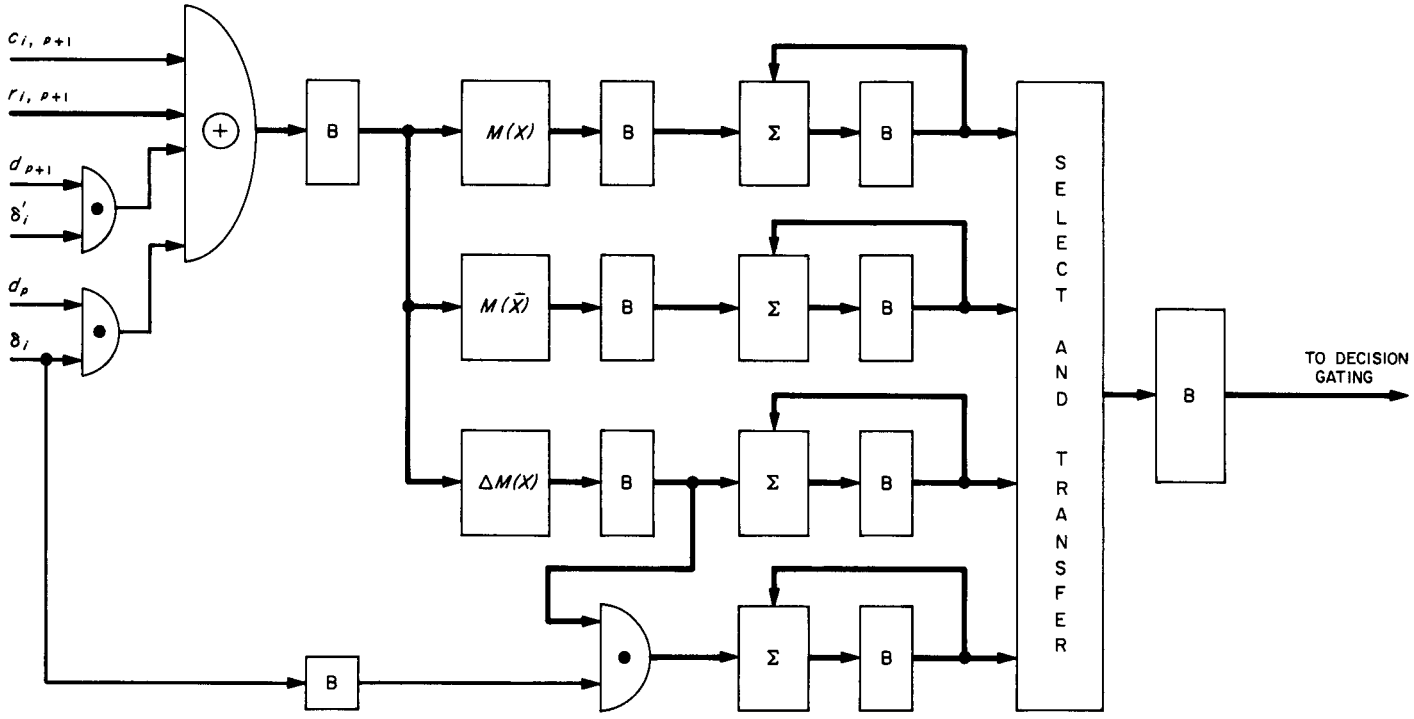

Fig. 36. Simplified block diagram of decoder arithmetic unit

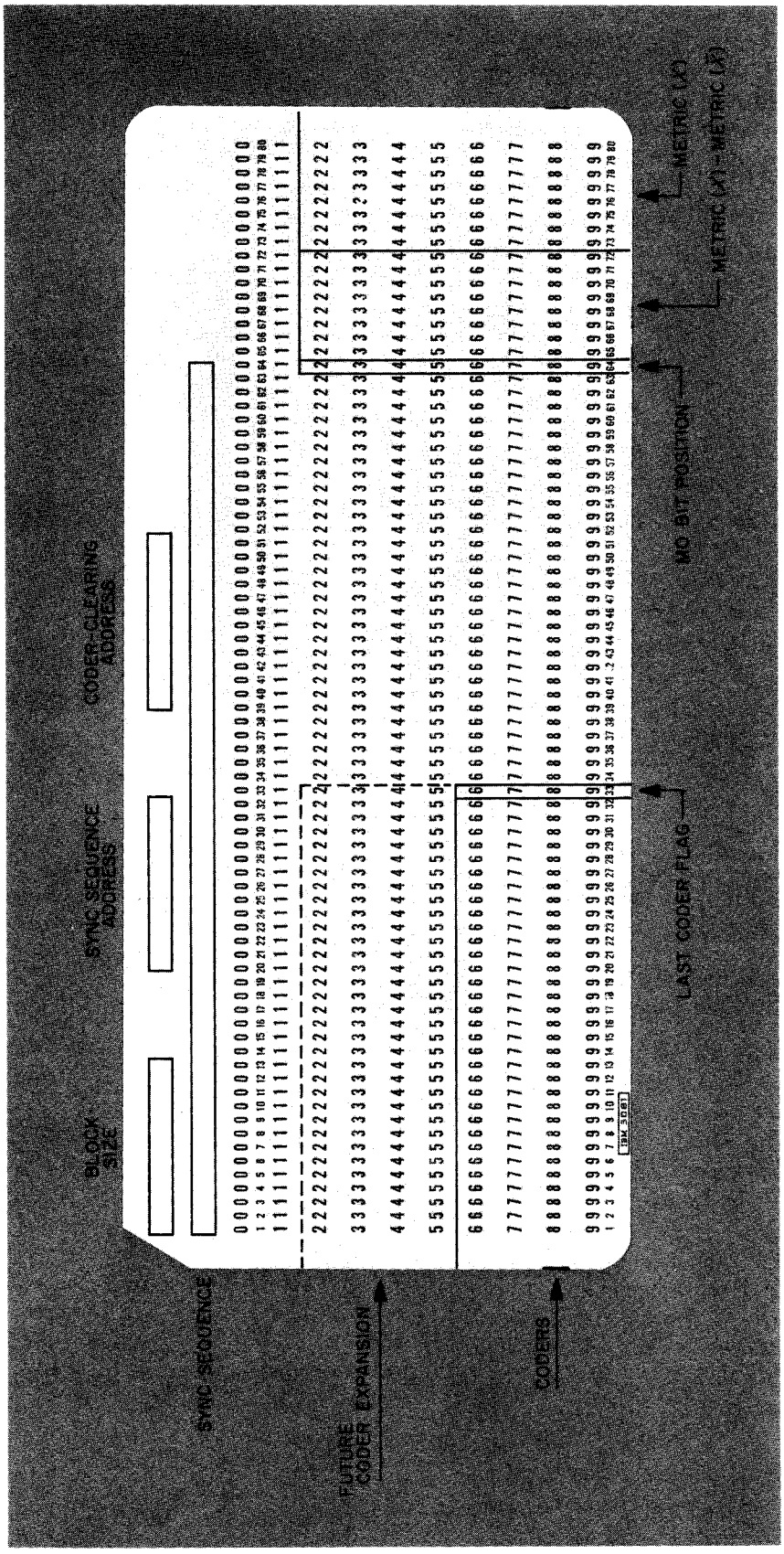

Fig. 37. Possible program card format

A simplified diagram of the decoder arithmetic unit is shown in Fig. 36. The intermediate buffering shown serves to reduce the amount of time a given element of the arithmetic unit stands idle and, hence, to increase the effective computation speed. For a code with V symbols per branch, the time from the initiation of a forward step to the completion of the succeeding computation phase is estimated to be $4 + V$ minor cycles, and the subsequent decoding may take 1 to 3 cycles, depending upon whether or not the primed metrics must be generated. A minor cycle should fall somewhere in the range 100 to 150 ns.

The static card reader (see Fig. 35) is an essential element in providing the desired decoder flexibility. As a simple, changeable, read-only memory, it defines such things as the coder taps, block size, the sync or forcing sequence, and the metric function. One of the possible card formats is shown in Fig. 37. While the present decoder is to be assembled for a maximum of four check symbols, or a V of five symbols per branch, little redesign would be necessary to expand this to perhaps $V = 9$ other than expanding the gating on the coder segment of the static card reader.

References

1. Jacobs, I. M., "Sequential Decoding for Efficient Communication from Deep Space," Paper 19TP67-937, *IEEE Trans. Commun. Technol.*, Vol. 15, No. 4, pp. 492–501, Aug. 1967.

2. Wozencraft, J. M., and Jacobs, I. M., *Principles of Communication Engineering*. John Wiley & Sons, Inc., New York, 1965.

6. Subroutines to Permit SDS FORTRAN II to Handle Magnetic Tapes With Arbitrary Formats, *W. B. Kendall*

a. Introduction. It is frequently desirable to use SDS FORTRAN II computer programs to process data recorded in some nonstandard format. To do this, it is necessary to have special subroutines to handle magnetic tapes, since the magnetic-tape handling routines of FORTRAN require tapes in a specific format. Furthermore, FORTRAN is basically a "one-file" system; it has no provisions for skipping files or taking automatic action when an end-of-file (EOF) mark is encountered on magnetic tape. (It does, however, have the ability to write an EOF.) For these reasons, a set of subroutines has been written to enable a FORTRAN program to handle magnetic tapes in arbitrary formats.

b. The subroutines. There are three separate subroutines in the set. These are as follows:

(1) A magnetic tape reading routine (TAPE).

(2) A routine for skipping records (or files), either forward or backward (SPACE).

(3) One of the standard SDS magnetic tape handlers (MTAPE—catalog numbers 040004 or 540001).

All actual tape movement is done by MTAPE. The TAPE and SPACE routines simply provide for communication with FORTRAN and, when necessary, for format conversion.

c. Use of the TAPE subroutine. One record from a magnetic tape can be read by the FORTRAN statement

```
CALL TAPE (IWDCNT, LOC, MODE,
IUNIT, ICHR, IERR, IOCNT)
```

where the arguments have the following significance.

IWDCNT is the number of words to be read from the record. If the record contains more than IWDCNT words, all extra words will be discarded. If fewer than IWDCNT words are in the record, then only the number of words actually in the record will be read. This will be indicated by the value returned in IOCNT.

LOC is an array to receive the words read.

MODE is a number that indicates whether the tape-read operation is to be in binary or binary-coded decimal mode. If MODE is even (i.e., its least-significant bit is zero), then the read will be in binary-coded decimal mode; otherwise, it will be in binary mode.

IUNIT is the tape unit number. If this is from 10 through 17 (decimal), then unit 0 through 7, respectively, on channel Y will be used. Otherwise, the least significant three bits of IUNIT are used as a unit number for channel W. In other words, if IUNIT is 0 through 7, then units 0 through 7 on channel W will be used.

ICHR is the number of characters per word on the tape. This can be from one through four. If it is less than four for a binary-coded decimal read, the characters read will be right justified in the computer words and the unfilled portions of the words will be unpredictable. (These right-justified words can easily be output in FORTRAN by using the R-format.) If ICHR is one or two for a binary read, the characters will be right justified with the sign bit (most significant bit read) extended to fill the left end of the words (i.e., standard two's complement form). Three-characters-per-word for a binary read is a special case. This is the format produced by JPL's analog-tape